Agent Reward Shaping for Alleviating Traffic Congestion

Kagan Tumer
NASA Ames Research Center
Mailstop 269-4
Moffett Field, CA 94035, USA
ktumer@mail.arc.nasa.gov

Adrian Agogino
UCSC, NASA Ames Research Center
Mailstop 269-3
Moffett Field, CA 94035, USA
adrian@email.arc.nasa.gov

ABSTRACT

Traffic congestion problems provide a unique environment to study how multi-agent systems promote desired system level behavior. What is particularly interesting in this class of problems is that no individual action is intrinsically "bad" for the system but that combinations of actions among agents lead to undesirable outcomes. As a consequence, agents need to learn how to coordinate their actions with those of other agents, rather than learn a particular set of "good" actions. This problem is ubiquitous in various traffic problems, including selecting departure times for commuters, routes for airlines, and paths for data routers.

In this paper we present a multi-agent approach to two traffic problems, where for each driver, an agent selects the most suitable action using reinforcement learning. The agent rewards are based on concepts from collectives and aim to provide the agents with rewards that are both easy to learn and that if learned, lead to good system level behavior. In the first problem, we study how agents learn the best departure times of drivers in a daily commuting environment and how following those departure times alleviates congestion. In the second problem, we study how agents learn to select desirable routes to improve traffic flow and minimize delays for all drivers. In both sets of experiments, agents using collective-based rewards produced near optimal performance (93-96% of optimal) whereas agents using system rewards (63-68%) barely outperformed random action selection (62-64%) and agents using local rewards (48-72%) performed worse than random in some instances.

1. INTRODUCTION

Multi-agent learning algorithms provide a natural approach to addressing congestion problems in traffic and transportation domains. Congestion problems are characterized by having the system performance depend on the number of agents that select a particular action, rather on the intrinsic value of those actions. Examples of such problems include lane/route selection in traffic flow [7, 10], path selection in data routing [8], and side selection in the minority game [3, 6]. In those problems, the desirability of lanes, paths or sides depends solely on the number of agents having selected them. Hence, multi-agent approaches that focus on agent coordination are ideally suited for these domains where agent coordination is critical for achieving desirable system behavior.

In this paper we apply multi-agent learning algorithms to two separate traffic problems. First we investigate how to coordinate the departure times of a set of drivers so that they do not end up producing traffic "spikes" at certain times, both providing delays at those times and causing congestion for future departures. In this problem, different time slots have different desirabilities that reflect user preferences for particular time slots. The system objective is to maximize the overall system's satisfaction as a weighted average of those desirabilities. In the second problem we investigate route selection where a set of drivers need to select different routes to a destination. In this problem, different routes have different capacities and the problem is for the agents to minimize the total congestion. Both problems share the same underlying property that agents greedily pursuing the best interests of their own drivers cause traffic to worsen for everyone in the system, including themselves.

The approach we present to alleviating congestion in traffic is based on assigning each driver an agent which determines the departure time/route to follow. Those agents determine their actions based on a reinforcement learning algorithm [9, 14, 18]. The key issue in this approach is to ensure that the agents receive rewards that promote good system level behavior. To that end, it is imperative that the agent rewards: (i) are aligned with the system reward[1], ensuring that when agents aim to maximize their own reward they also aim to maximize system reward; and (ii) are sensitive to the actions of the agents, so that the agents can determine the proper actions to select (i.e., they need to limit the impact of other agents in the reward functions of a particular agent).

The difficulty in agent reward selection stems from the fact that typically these two properties provide conflicting requirements. A reward that is aligned with the system reward usually accounts for the actions of other agents, and thus is likely to not be sensitive to the actions of one agent; on the other hand, a reward that is sensitive to the actions of one agent is likely not to be aligned with system reward.

[1]We call the function rating the performance of the full system, "system reward" throughout this paper in order to emphasize its relationship to agent rewards.

AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.

This issue is central to achieving coordination in a traffic congestion problem and has been investigated in various fields such as computational economics, mechanism design, computational ecologies and game theory [2, 12, 5, 11, 13]. We address this reward design problem using the difference reward derived from collectives [19, 16], which provides a good balance of alignedness and sensitivity. The difference reward has been applied to many domains, including rover coordination [1], faulty device selection problem [15], packet routing over a data network [17, 20], and modeling nongenomic models of early life [**?**].

In this paper we show how these collective based reinforcement learning methods can be used to alleviate traffic congestion. In Section 2 we discuss the properties agent rewards need to have and present a particular example of agent reward. In Sections 3.1 and 3.2 we present the departure coordination problem. The results in this domain show that total traffic delays can be improved significantly when agents use collective based rewards. In Section 3.3 we present the route selection problem. The results in this domain show that traffic congestion can be reduced by over 30% when agents use collective based rewards. Finally Section 4 we discuss the implication of these results and discuss methods by which they can be applied in the traffic domain.

2. BACKGROUND

In this work, we focus on multi-agent systems where each agent, i, tries to maximize its reward function $g_i(z)$, where z depends on the joint move of all agents. Furthermore, there is a system reward function, $G(z)$ which rates the performance of the full system. To distinguish states that are impacted by actions of agent i, we decompose[2] z into $z = z_i + z_{-i}$, where z_i refers to the parts of z that are dependent on the actions of i, and z_{-i} refers to the components of z that do not depend on the actions of agent i.

2.1 Properties of Reward Functions

Now, let us formalize the two requirements discussed above that an agent's reward should satisfy in order for the system to display coordinated behavior . First, the agent rewards have to be aligned with respect to G, quantifying the concept that an action taken by an agent that improves its own reward also improves the system reward. Formally, for systems with discrete states, the degree of **factoredness** for a given reward function g_i is defined as:

$$\mathcal{F}_{g_i} = \frac{\sum_z \sum_{z'} u[(g_i(z) - g_i(z'))\ (G(z) - G(z'))]}{\sum_z \sum_{z'} 1} \qquad (1)$$

for all z' such that $z_{-i} = z'_{-i}$ and where $u[x]$ is the unit step function, equal to 1 if $x > 0$, and zero otherwise. Intuitively, the higher the degree of factoredness between two rewards, the more likely it is that a change of state will have the same impact on the two rewards. A system is fully factored when $\mathcal{F}_{g_i} = 1$.

Second, an agent's reward has to be sensitive to its own actions and insensitive to actions of others. Formally we can quantify the **learnability** of reward g_i, for agent i at z:

$$\lambda_{i,g_i}(z) = \frac{E_{z'_i}[|g_i(z) - g_i(z_{-i} + z'_i)|]}{E_{z'_{-i}}[|g_i(z) - g_i(z'_{-i} + z_i)|]} \qquad (2)$$

where $E[\cdot]$ is the expectation operator, z'_i's are alternative actions of agent i at z, and z'_{-i}'s are alternative joint actions of all agents other than i. Intuitively, learnability provides the ratio of the expected value of g_i over variations in agent i's actions to the expected value of g_i over variations in the actions of agents other than i. So at a given state z, the higher the learnability, the more $g_i(z)$ depends on the move of agent i, i.e., the better the associated signal-to-noise ratio for i. Higher learnability means it is easier for i to achieve large values of its reward.

2.2 Difference Reward Functions

Let us now focus on providing agent rewards that are both high factoredness and high learnability. Consider the **difference** reward [19], which is of the form:

$$D_i \equiv G(z) - G(z_{-i} + c_i) \qquad (3)$$

where z_{-i} contains all the states on which agent i has no effect, and c_i is a fixed vector. In other words, all the components of z that are affected by agent i are replaced with the fixed vector c_i. Such difference reward functions are fully factored no matter what the choice of c_i, because the second term does not depend on i's states [19]. Furthermore, they usually have far better learnability than does a system reward function, because the second term of D removes some of the effect of other agents (i.e., noise) from i's reward function. In many situations it is possible to use a c_i that is equivalent to taking agent i out of the system. Intuitively this causes the second term of the difference reward function to evaluate the value of the system without i and therefore D evaluates the agent's contribution to the system reward.

The difference reward can be applied to any linear or nonlinear system reward function. However, its effectiveness is dependent on the domain and the interaction among the agent reward functions. At best, it fully cancels the effect of all other agents. At worst, it reduces to the system reward function, unable to remove any terms (e.g., when z_{-i} is empty, meaning that agent i effects all states). In most real world applications, it falls somewhere in between, and has been successfully used in many domains including agent coordination, satellite control, data routing, job scheduling and congestion games [1, 17, 19]. Also note that computationally the difference reward is often easier to compute than the system reward function [17]. Indeed in the problem presented in this paper, for agent i, D_i is easier to compute than G is (see details in Section 3.1.1).

2.3 Reward Maximization

In this paper we assume that each agent maximize its own reward using its own reinforcement learner (though alternatives such as evolving neuro-controllers are also effective [1]. For complex delayed-reward problems, relatively sophisticated reinforcement learning systems such as temporal difference may have to be used. However, the traffic domain modeled in this paper only needs to utilize immediate rewards, therefore a simple table-based immediate reward reinforcement learning is used. Our reinforcement learner is equivalent to an ϵ-greedy Q-learner with a discount rate of

[2]Instead of concatenating partial states to obtain the full state vector, we use zero-padding for the missing elements in the partial state vector. This allows us to use addition and subtraction operators when merging components of different states (e.g., $z = z_i + z_{-i}$).

0. At every episode an agent takes an action and then receives a reward evaluating that action. After taking action a and receiving reward R a driver updates its table as follows: $Q'(a) = (1-\alpha)Q(a)+\alpha(R)$, where α is the learning rate. At every time step the driver chooses the action with the highest table value with probability $1-\epsilon$ and chooses a random action with probability ϵ. In the experiments described in the following section, α is equal to 0.5 and ϵ is equal to 0.05. The parameters were chosen experimentally, though system performance was not overly sensitive to these parameters.

3. EXPERIMENTS

To test the effectiveness of our rewards in the traffic congestion domain, we performed experiments using two abstract traffic models. In the first model each agent has to select a time slot to start its drive. In this model we explore both simple and cascading traffic flow. With non-cascading flow, drivers enter and exit the same time slot, while with cascading flow, drivers stuck in a time slot with too many other drivers stay on the road for future time slots. In the second model, instead of choosing time slots, drivers choose routes. In this model the system reward has different properties and we have the additional complexity of different routes having different capacities.

3.1 Single-Route Congestion Model

In the traffic congestion model used here, there is a fixed set of drivers, driving on a single route. The agents choose the time slot in which their drivers start their commutes. The system reward is given by:

$$G = \sum_t w_t S(k_t) \ . \tag{4}$$

where weights w_t model rush-hour scenarios where different time slots have different desirabilities, and $S(k)$ is a "time slot reward", depending on the number of agents that chose to depart in the time slot:

$$S(k) = \begin{cases} ke^{-1} & \text{if } k \leq c \\ ke^{-k/c} & \text{otherwise} \end{cases} \ , \tag{5}$$

The number of drivers in the time slot is given by k, and the optimal capacity of the time slot is given by c. Below an optimal capacity value c, the reward of the time slot increases linearly with the number of drivers. When the number of drivers is above the optimal capacity level, the value of the time slot decreases quickly (asymptotically exponential) with the number of drivers. This reward models how drivers do not particularly care how much traffic is on a road until it is congested. This function is shown in Figure 1. In this problem the task of the system designer is to have the agents choose time slots that help maximize the system reward. To that end, agents have to balance the benefit of going at preferred time slots with the congestion at those time slots.

3.1.1 Driver Rewards

While as a system designer our goal is to maximize the system reward, we have each individual agent try to maximize a driver-specific reward that we select. The agents maximize their rewards through reinforcement learning, where they learn to choose time slots that have expected high reward. In these experiments, we evaluate the effectiveness

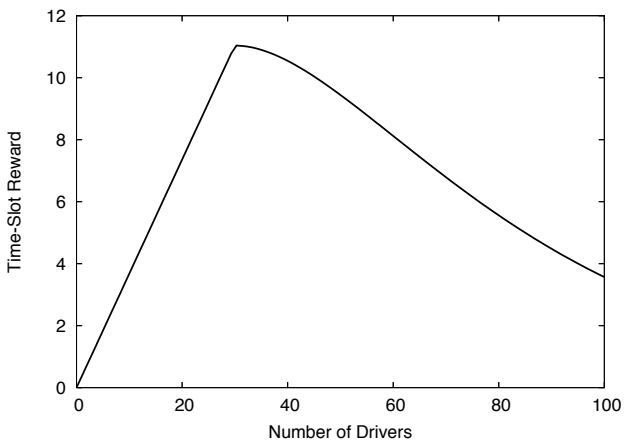

Figure 1: Reward of time slot with $c = 30$.

of three different rewards. The first reward is simply the system reward G, where each agent tries to maximize the system reward directly. The second reward is a local reward, L_i where each agent tries to maximize a reward based on the time slot it selected:

$$L_i(k) = w_i S(k_i) \ , \tag{6}$$

where k_i is the number of drivers in the time slot chosen by driver i. The final reward is the difference reward, D:

$$\begin{aligned} D_i &= G(k) - G(k_{-i}) \\ &= \sum_j L_j(k) - \sum_j L_j(k_{-i}) \\ &= L_i(k) - L_i(k_{-i}) \\ &= w_i k_i S(k_i) - w_i(k_i - 1)S(k_i - 1) \ , \end{aligned}$$

where k_{-i} represents the the driver counts when driver i is taken out of the system. Note that since taking away driver i only affects one time slot, all of the terms but one cancel out, making the difference reward simpler to compute than the system reward.

3.1.2 Results

In this set of experiments there were 1000 drivers, and the optimal capacity of each time slot was 250. Furthermore, the weighting vector was centered at the most desirable time slot (e.g., 5 PM departures):

$$w = [1\ 5\ 10\ 15\ 20\ 15\ 10\ 5\ 1]^T \ .$$

This weighting vector reflects a preference for starting a commute at the end of the workday with the desirability of a time slot decreasing for earlier and later times.

This experiment shows that drivers using the difference reward are able to quickly obtain near-optimal system performance (see Figure 2). In contrast, drivers that try to directly maximize the system reward learn very slowly and never achieve good performance during the time-frame of the experiment. This slow learning rate is a result of the system reward having low learnability to the agents' actions. Even if a driver were to take a system wide coordinated action, it is likely that some of the 999 other drivers would take uncoordinated actions at the same time, lowering the value of the

system reward. A driver using the system reward typically does not get proper credit assignment for its actions, since the reward is dominated by other drivers.

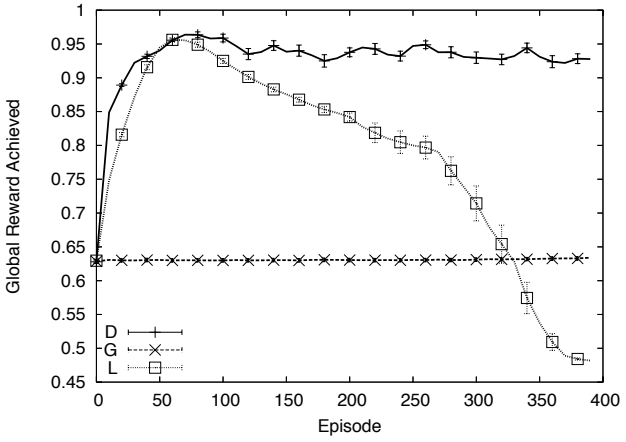

Figure 2: Performance on Single-Route Domain. Drivers using difference reward quickly learn to achieve near optimal performance (1.0). Drivers using system reward learn slowly. Drivers using non-factored local reward eventually learn counterproductive actions.

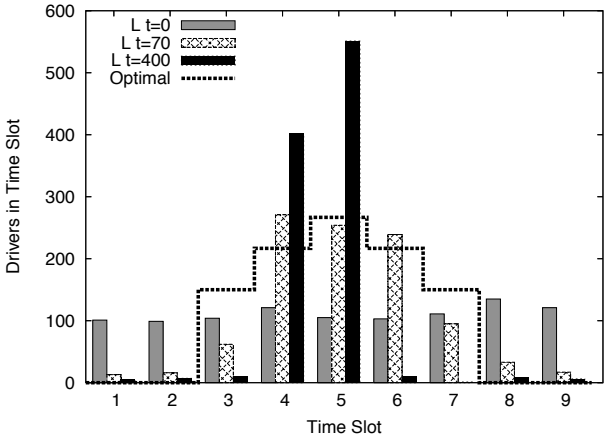

Figure 3: Distribution of Drivers using Local Reward. Early in training drivers learn good policies. Later in learning, the maximization of local reward causes drivers to over utilize high valued time slots.

The experiment where drivers are using L (a non-factored local reward) exhibit some interesting performance properties. At first these drivers learn to improve the system reward. However, after about episode seventy their performance starts to decline. Figure 3 gives greater insight into this phenomenon. At the beginning of the experiment, the drivers are randomly distributed among time slots, resulting in a low reward. Later in training agents begin to learn to use the time slots that have the most benefit. When the number of drivers reach near optimal values for those time slots, the system reward is high. However, all agents in the system covet those time slots and more agents start to select the desirable time slots. This causes congestion and system reward starts to decline. This performance characteristics is typical of system with agent rewards of low factoredness. In such a case, agents attempting to maximize their own rewards lead to undesirable system behavior. In contrast, because their rewards are factored with the system reward, agents using the difference reward form a distribution that more closely matches the optimal distribution (Figure 4).

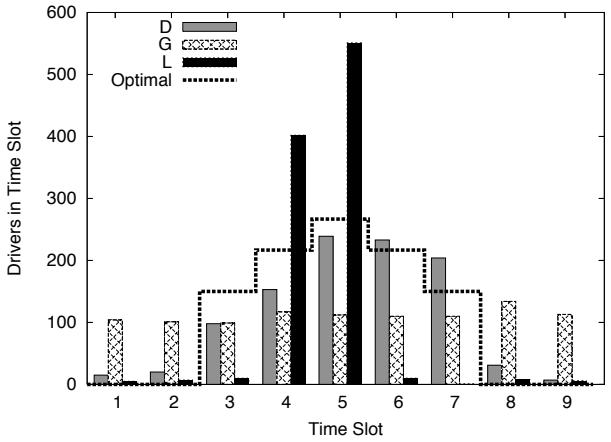

Figure 4: Distribution of Drivers at end of Training. Drivers using difference reward form distribution that is closer to optimal than drivers using system of local rewards.

3.2 Cascading Single-Route Congestion Model

The previous single-route model assumes that drivers enter and leave the same time slot. Here we introduce a more complex model, where drivers remain in the system longer when it is congested. This property modeled by having drivers over the optimal capacity, c stay in the system until they reach a time slot with a traffic level below c. When the number of drivers in a time slot is less than c the reward for a time slot is the same as before. When the number of drivers is above c the linear term k is replaced with c:

$$S(k) = \begin{cases} ke^{-1} & \text{if } k \leq c \\ ce^{-k/c} & \text{otherwise} \end{cases} \quad (7)$$

As before the system reward is a sum of the time slot rewards: $G = \sum_t S(k_t)$.

3.2.1 Driver Rewards

Again the local reward is the weighted time slot reward:

$$L_i = w_i S(k_i) \,, \quad (8)$$

where k_i is the number of drivers in the time slot chosen by driver i. However the difference reward is more difficult to simplify as the actions of a driver can have influence over several time slots:

$$\begin{aligned} D_i &= G(k) - G(k_{-i}) \\ &= \sum_j w_j S(k_j) - \sum_j w_j S(k_{-i_j}) \,, \end{aligned}$$

where k_{-i_j} is the number of drivers there would have been in time slot j had driver i not been in the system.

3.2.2 Results

Figure 5 shows the results for cascading traffic model. As previously, there are 1000 drivers and time slot capacities are 250. Drivers using the different rewards exhibit similar characteristics on this model than on the non-cascading one. Again drivers using the system reward are unable to improve their performance significantly beyond their initial random performance. In this model drivers using the local reward perform even worse once they become proficient at maximizing their own reward. The local reward here performs worse, because in this model a driver's choice of time slot can cause additional side-effects for other time slots, as drivers from a congested time slot remain in the system for future time slots. As a result, when drivers using the local reward cause congestion for their time slots, the congestion cascades as drivers spill into future time slots causing a significant decrease in performance.

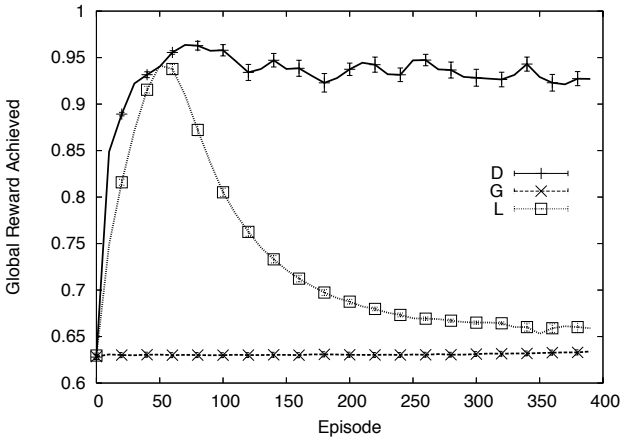

Figure 5: Performance on Cascading Single-Route Domain. In this domain drivers above the capacity in one time slot remain in system in future time slots. Drivers using difference reward quickly learn to achieve near optimal performance (1.0).

3.3 Multiple-Route Congestion Model

In this model instead of selecting time slots, drivers select routes. The main difference in this model is the functional form of the reward for a route as shown in Figure 6. In this model the objective is to keep the routes uncongested. The system reward does not care how many drivers are on a particular route as long as that route is below its congestion point. Each route has a different weight representing overall driver preference for a route. Furthermore, each route has its own capacity, modeling the realities that some routes having more lanes than others.

In this model the reward for an individual route is:

$$S(k,c) = \begin{cases} e^{-1} & \text{if } k \leq c \\ e^{-k/c} & \text{otherwise} \end{cases} \quad (9)$$

The system reward is then the sum of all route rewards

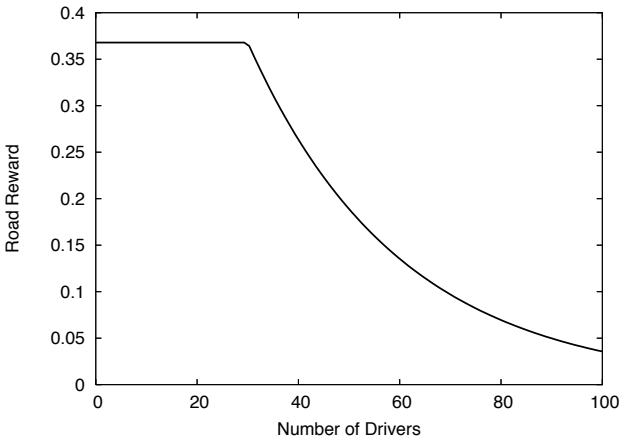

Figure 6: Reward of Road with $c = 30$.

weighted by the value of the route.

$$G = \sum_i w_i S(k_i, c_i) \ , \quad (10)$$

where w_i is the weighting for route i and c_i is the capacity for route i.

3.3.1 Driver Rewards

Again three rewards were tested: the system reward, the local reward and the difference reward. The local reward is the weighted reward for a single route:

$$L_i = w_i S(k_i, c_i) \ . \quad (11)$$

The final reward is the difference reward, D:

$$\begin{aligned} D_i &= G(k) - G(k_{-i}) \\ &= L_i(k) - L_i(k_{-i}) \\ &= w_i S(k_i, c_i) - w_i S(k_i - 1, c_i) \ , \end{aligned}$$

representing the difference between the actual system reward and what the system reward would have been if the driver had not been in the system.

3.3.2 Results

Here we show the results of experiments where we test performance of the three rewards in the multi-route model, where different routes have different value weightings and different capacities. There were 1000 drivers in these experiments and the route capacities were 333, 167, 83, 33, 17, 33, 83, 167, 333. Each route is weighted with the weights 1, 5, 10, 1, 5, 10, 1, 5, 10. Figure 7 shows that drivers using the system reward perform poorly, and learn slowly. Again drivers using the difference reward perform the best, learning quickly to achieve an almost optimal solution. Drivers using the local reward learn more quickly early in training than drivers using the system reward, but never achieve as high as performance as those using the difference reward. However in this domain the drivers using the local reward do not degrade from their maximal performance, but instead enter a steady state that is significantly below that of the drivers using the difference reward.

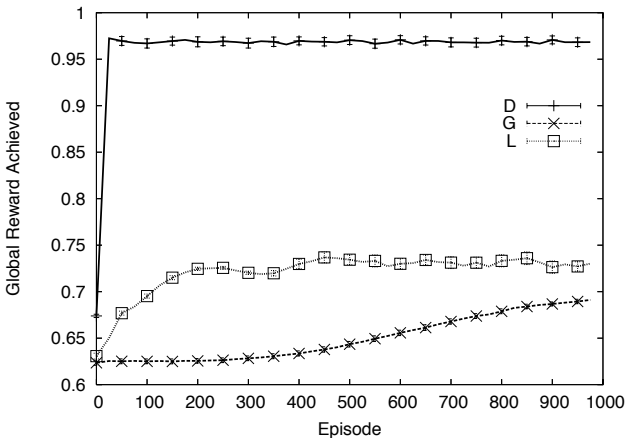

Figure 7: Performance on Domain with Multiple Routes. Best observed performance = 1.0 (optimal not calculated)

4. DISCUSSION

This paper presented a method for improving congestion in two different traffic problems. First we presented a method by which agents can coordinate the departure times of drivers in order to alleviate spiking at peak traffic times. Second we showed that agents can manage effective route selection and significantly reduce congestion by using a reward structure that penalizes greedily pursuing the routes with high capacity. Both results are based on agents receiving rewards that have high factoredness and high learnability (i.e., are both aligned with the system reward and are as sensitive as possible to changes in the reward of each agent). In both sets of experiments, agents using collective-based rewards produced near optimal performance (93-96% of optimal) whereas agents using system rewards (63-68%) barely outperformed random action selection (62-64%) and agents using local rewards (48-72%) provided performance ranging from mediocre to worse than random in some instances.

One issue that arises in traffic problems that does not arise in many other domains (e.g., rover coordination) is in ensuring that drivers follow the advice of their agents. In this work, we did not address this issue, as our purpose was to show that solutions to the difficult traffic congestion problem can be addressed in a distributed adaptive manner using intelligent agents. Ensuring that drivers follow the advice of their agents is a fundamentally different problem. On one hand, drivers will notice that the departure times/routes suggested by their agents provide significant improvement over their regular patterns. However, as formulated, there are no mechanisms for ensuring that a driver does not gain an advantage by ignoring the advice of his or her agent.

5. REFERENCES

[1] A. Agogino and K. Tumer. Unifying temporal and structural credit assignment problems. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 978–985, New York, NY, July 2004.

[2] C. Boutilier. Planning, learning and coordination in multiagent decision processes. In *Proceedings of the Sixth Conference on Theoretical Aspects of Rationality and Knowledge*, Holland, 1996.

[3] D. Challet and Y. C. Zhang. On the minority game: Analytical and numerical studies. *Physica A*, 256:514, 1998.

[4] N. Gupta, A. Agogino, and K. Tumer. Efficient agent-based models for non-genomic evolution. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multi-Agent Systems*, Hakodate, Japan, May 2006. to appear.

[5] B. A. Huberman and T. Hogg. The behavior of computational ecologies. In *The Ecology of Computation*, pages 77–115. North-Holland, 1988.

[6] P. Jefferies, M. L. Hart, and N. F. Johnson. Deterministic dynamics in the minority game. *Physical Review E*, 65 (016105), 2002.

[7] B. S. Kerner and H. Rehborn. Experimental properties of complexity in traffic flow. *Physical Review E*, 53(5):R4275–4278, 1996.

[8] A. A. Lazar, A. Orda, and D. E. Pendarakis. Capacity allocation under noncooperative routing. *IEEE Transactions on Networking*, 5(6):861–871, 1997.

[9] M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th International Conference on Machine Learning*, pages 157–163, 1994.

[10] K. Naigel. Experiences with iterated traffic microsimulations in dallas. pre-print adap-org/9712001, December 1997.

[11] D. C. Parkes. *Iterative Combinatorial Auctions: Theory and Practice*. PhD thesis, University of Pennsylvania, 2001.

[12] T. Sandholm and R. Crites. Multiagent reinforcement learning in the iterated prisoner's dilemma. *Biosystems*, 37:147–166, 1995.

[13] P. Stone and M. Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3), 2000.

[14] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[15] K. Tumer. Designing agent utilities for coordinated, scalable and robust multi-agent systems. In P. Scerri, R. Mailler, and R. Vincent, editors, *Challenges in the Coordination of Large Scale Multiagent Systems*, pages 173–188. Springer, 2005.

[16] K. Tumer and D. Wolpert, editors. *Collectives and the Design of Complex Systems*. Springer, New York, 2004.

[17] K. Tumer and D. H. Wolpert. Collective intelligence and Braess' paradox. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, pages 104–109, Austin, TX, 2000.

[18] C. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3/4):279–292, 1992.

[19] D. H. Wolpert and K. Tumer. Optimal payoff functions for members of collectives. *Advances in Complex Systems*, 4(2/3):265–279, 2001.

[20] D. H. Wolpert, K. Tumer, and J. Frank. Using collective intelligence to route internet traffic. In *Advances in Neural Information Processing Systems -*

11, pages 952–958. MIT Press, 1999.